ENSF-381: Assignment 4

Winter 2025

Department of Electrical and Software Engineering Schulich

School of Engineering

Due: March 28, 2025, at 11:59 PM

Objective

In this assignment, you will build the Learning Management System (LMS) using React. The LMS will allow students to explore courses, enroll in them, and view testimonials. You will create reusable components, manage state, and implement dynamic interactions while adhering to React best practices.

Administrative Details

- The submission should be a ***single compressed file (.zip)*** that includes the `src` folder and resources (e.g., images), ready for execution without modifications.
- Name the submitted file as Assignment4_ENSF381_Section#_Group# Example: `Assignment4_ENSF381_L02_Group01.zip`
- Submit ONLY ONE copy.
- Include group members' names and UCIDs as comments in `index.js`.
- *Do not modify component names* or add extra styling/functionality beyond requirements

---

1. Project Setup (10 Marks)

A. Create React App:

```
npx create-react-app my-lms-app
cd my-lms-app
```

B. File Structure:

a. Create `components` and `data` directories inside `src`.
b. Add `courses.js` and `testimonials.js` to `src/data`. (See the data files below)
c. Create `src/images` and add course thumbnails (e.g., `course1.jpg`, `course2.jpg`).

C. Install Dependencies:

```
npm install react-router-dom
```

List all dependencies in readme.txt with installation commands.

Data Files

1. courses.js:

```
const courses = [
  {
    id: 1,
    name: "Web Development",
    instructor: "Dr. John Smith",
    description: "Master HTML, CSS, and JavaScript.",
    duration: "8 weeks",
    image: "images/course1.jpg"
  },
  // Add 9 more courses...
];
export default courses;
```

2. testimonials.js

```
const testimonials = [
  {
    studentName: "Alice Johnson",
    courseName: "Web Development",
    review: "Excellent course structure!",
    rating: 5
  },
  // Add 3 more testimonials...
];
export default testimonials;
```

---

2. Homepage Requirements (30 Marks)

Components:

A. `Header` *Component:*

a. Display LMS logo (left) and navigation links (Home, Courses, Login) using React Router.

b. Style links with equal spacing.

B. `MainSection` *Component:*

a. About LMS: Brief description of the system.

b. Featured Courses: Display 3 random courses from `courses.js`.

c. Testimonials: Show 2 random testimonials from `testimonials.js` on each render (use `useEffect`). Include student name, review, and star rating (e.g., ★★★★☆).

C. `Footer` *Component:*

a. Copyright notice at the bottom.

Structure of Homepage.js

```
<div>
  <Header />
  <MainSection />
  <Footer />
</div>
```

---

3. Courses Page Requirements (40 Marks)

Components

A. `CourseItem` *Component:*

a. Displays course image, name, instructor, and "`Enroll Now`" button.
b. Hover Effect: Show course description on `onMouseEnter` (use `useState`).

B. `EnrolledCourse` *Component:*

a. Represents an enrolled course in the Enrollment List.
b. Displays course name, credit hours, and a "`Drop Course`" button.
c. Clicking "`Drop Course`" decreases enrollment count; removes course if count hits 0.

C. `EnrollmentList` *Component:*

a. Lists all enrolled courses using `EnrolledCourse`.
b. Calculates and displays total credit hours (sum of all enrolled courses).
c. Uses `useEffect` to save/load enrollment data from local storage.

D. `CourseCatalog` *Component:*

a. Renders all courses from `courses.js` using `CourseItem`.

Structure of CoursesPage.js

```
<div className="courses-page">
  <Header />
  <div className="content">
    <CourseCatalog />
    <EnrollmentList />
  </div>
  <Footer />
</div>
```

---

4. Login Page Requirements (20 Marks)

Components

A. `LoginForm` *Component:*

a. UI Elements:

- Username and Password input fields.
- `Login` button.
- Forgot Password link (non-functional).

b. Validate inputs:

- Username and password cannot be empty.
- Password must be at least 8 characters.

c. API Integration:

- Fetch user data from `https://jsonplaceholder.typicode.com/users`.
- Validate credentials against the API's username and email fields.

d. Dynamic Feedback:

- Display success/error messages in a styled `<div>`.
- Redirect to /courses on successful login after 2 seconds.

c. Implementation Details

- Use `useState` for form state management.
- Use `useEffect` to handle API calls and redirects.
- Style the form to match the LMS theme (reuse CSS classes from Homepage).

B. `AuthMessage` *Component:*

a. Implement global state management for user authentication using `useContext` hook.

- Create the context and context provider in `LoginForm.js`.
- Consume the username and password contexts in `AuthMessage.js`.

b. Display styled success/error messages from `DisplayStatus.js` component.

c. Use the following props to render the `DisplayStatus.js` component:

- `type`: "success" or "error".
- `message`: Text to display.

C. `DisplayStatus.js` *Component:*

a. Use the `type` and `message` props to display the login message.

Structure of LoginForm.js

```
<div>
  <Header />
  <LoginForm />
  <Footer />
</div>
```
